Guide Utilisateur COBOL to Java Translator

Projet COBOL Translator

02/01/2026

Table des matières

1 Guide Utilisateur Complet

1.1 COBOL to Java Spring Batch Translator

Version : 1.0.0 **Date** : 2026-01-02 **Auteur** : Projet COBOL Translator

---

1.2 Table des matières

1.3 1. Introduction

1.3.1 1.1 Qu'est-ce que COBOL Translator ?

COBOL Translator est un outil professionnel qui convertit automatiquement vos programmes COBOL en projets Java Spring Batch modernes. Il utilise :

- **ANTLR4** pour le parsing syntaxique précis
- **Abstract Syntax Tree (AST)** pour l'analyse structurelle
- **Générateurs de code** pour créer des projets Spring Batch complets

1.3.2 1.2 Fonctionnalités principales

Parsing COBOL avancé - Support de ~90% des constructions COBOL - Détection d'erreurs syntaxiques précises - Gestion des 4 divisions COBOL

Génération Java complète - Jobs Spring Batch configurés - Entités de données - Processeurs métier - Configuration Maven complète

Interface utilisateur - **CLI** : Ligne de commande pour scripts automatisés - **Web** : Interface graphique drag & drop

---

1.4 2. Prérequis

1.4.1 2.1 Configuration système requise

Composant	Version minimale	Recommandé
Java JDK	17	17 ou 21
Maven	3.6+	3.9+
RAM	2 GB	4 GB
Espace disque	500 MB	1 GB
Système d'exploitation	Linux, macOS, Windows	Tous

1.4.2 2.2 Vérifier les prérequis

Ouvrez un terminal et vérifiez :

```
# Vérifier Java
java -version
# Résultat attendu: openjdk version "17.x.x" ou supérieur

# Vérifier Maven
mvn -version
# Résultat attendu: Apache Maven 3.x.x
```

1.4.3 2.3 Installation de Java (si nécessaire)

Sur Ubuntu/Debian :

```
sudo apt update
sudo apt install openjdk-17-jdk
```

Sur macOS :

```
brew install openjdk@17
```

Sur Windows : - Télécharger depuis adoptium.net - Installer et ajouter au PATH

1.5 3. Installation

1.5.1 3.1 Récupération du projet

Si vous avez reçu une archive :

```
cd ~/Desktop
unzip cobol-to-java-translator.zip
cd cobol-to-java-translator
```

Si vous clonez depuis Git :

```
git clone https://github.com/your-org/cobol-to-java-translator.git
cd cobol-to-java-translator
```

1.5.2 3.2 Compilation du projet

```
# Compilation complète avec tests
mvn clean package

# OU compilation rapide sans tests
mvn clean package -DskipTests
```

Résultat attendu :

```
[INFO] BUILD SUCCESS
[INFO] Total time: 3.5 s
```

Le fichier JAR est créé : `target/cobol-translator.jar`

1.5.3 3.3 Vérification de l'installation

```
java -jar target/cobol-translator.jar --version
```

Résultat attendu :

```
COBOL to Java Translator v1.0.0
```

1.6 4. Démarrage des services

1.6.1 4.1 Service CLI (Ligne de commande)

Le service CLI est **toujours disponible** dès que le JAR est compilé.

Pas de démarrage requis - utilisez directement les commandes.

Vérification :

```
java -jar target/cobol-translator.jar --help
```

Résultat :

```
Usage: cobol-translator [-hV] [COMMAND]
Translates COBOL programs to Java Spring Batch
  -h, --help      Show this help message and exit.
  -V, --version   Print version information and exit.
Commands:
  translate      Translate a single COBOL file
  translate-all  Translate all COBOL files in a directory
```

1.6.2 4.2 Service Web (Interface graphique)

1.6.2.1 4.2.1 Démarrage du serveur web Commande de base :

```
java -jar target/cobol-translator.jar
```

OU avec Maven :

```
mvn spring-boot:run
```

1.6.2.2 4.2.2 Logs de démarrage Vous devriez voir :

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.0)

2026-01-02 10:00:00 INFO  o.s.b.w.e.tomcat.TomcatWebServer :
  Tomcat started on port(s): 9090 (http)
2026-01-02 10:00:00 INFO  c.c.t.CobolTranslatorApplication :
  Started CobolTranslatorApplication in 2.5 seconds
```

** Indicateurs de succès :** - Message `Started CobolTranslatorApplication` - Port `9090` mentionné - Aucune erreur rouge

1.6.2.3 4.2.3 Accès à l'interface web Ouvrez votre navigateur à l'adresse :

`http://localhost:9090/conversion`

Page attendue : - Header violet "COBOL to Java Spring Batch Converter" - Formulaire avec champs "Nom du projet" et "Package" - Zone de drag & drop pour fichiers

1.6.2.4 4.2.4 Arrêt du service web Dans le terminal où le service tourne :

`Ctrl + C`

Le service s'arrête proprement.

1.6.3 4.3 Service H2 Console (Base de données - optionnel)

La console H2 est **automatiquement démarrée** avec le service web.

Accès :

`http://localhost:9090/h2-console`

Connexion : - **JDBC URL** : `jdbc:h2:mem:translatordb` - **Username** : `sa` - **Password** : (laisser vide)

Utilité : Inspecter les métadonnées Spring Batch après exécution.

1.6.4 4.4 Résumé des URLs des services

Service	URL	État par défaut
Interface Web	http://localhost:9090/conversion	Actif
Console H2	http://localhost:9090/h2-console	Actif
CLI	N/A (commandes directes)	Toujours actif

1.7 5. Utilisation en ligne de commande

1.7.1 5.1 Conversion d'un fichier unique

Syntaxe :

```
java -jar target/cobol-translator.jar translate <fichier.cob> [options]
```

Exemple basique :

```
java -jar target/cobol-translator.jar translate examples/customer.cob
```

Avec options :

```
java -jar target/cobol-translator.jar translate examples/customer.cob \
  --package com.acme.batch \
  --output-dir ./generated-projects
```

1.7.2 5.2 Options disponibles

Option	Description	Défaut
`--package`	Package Java de base	com.generated.batch
`--output-dir`	Répertoire de sortie	. (répertoire courant)
`--generate-tests`	Générer les tests	true
`--generate-docs`	Générer la documentation	true

1.7.3 5.3 Conversion d'un répertoire entier

Syntaxe :

```
java -jar target/cobol-translator.jar translate-all <répertoire> [options]
```

Exemple :

```
java -jar target/cobol-translator.jar translate-all ./cobol-programs \
  --package com.company.migration \
  --output-dir ./java-projects
```

Résultat : Tous les fichiers `.cob` et `.cbl` du répertoire sont convertis.

1.7.4 5.4 Exemples de sortie CLI

Succès :

```
[INFO] Parsing COBOL file: customer.cob
[INFO] Generating Job configuration...
```

```
[INFO] Generating Entity classes...
[INFO] Generating Processor...
[INFO] Creating Maven project structure...
[SUCCESS] Conversion completed successfully!
[INFO] Output: ./customer-batch/
```

Erreur :

```
[ERROR] Failed to parse customer.cob
[ERROR] Syntax error at line 42:15 - unexpected token 'END'
```

1.8 6. Utilisation de l'interface web

1.8.1 6.1 Démarrage (rappel)

```
java -jar target/cobol-translator.jar
```

Puis accéder à : `http://localhost:9090/conversion`

1.8.2 6.2 Processus de conversion

1.8.2.1 Étape 1 : Remplir le formulaire Formulaire

Champs obligatoires : - **Nom du projet** : Nom du projet Spring Batch généré - Exemple : `customer-batch-migration` - Règles : lettres, chiffres, tirets uniquement

Champs optionnels : - **Package de base** : Package Java racine - Exemple : `com.company.customer.batch` - Défaut : `com.example.batch` - Règles : format Java valide (minuscules, points)

1.8.2.2 Étape 2 : Upload des fichiers COBOL **Méthode A : Cliquer** 1. Cliquez sur la zone " Cliquez ou glissez-déposez..." 2. Sélectionnez vos fichiers `.cob` ou `.cbl` 3. Validez la sélection

Méthode B : Glisser-déposer 1. Ouvrez votre explorateur de fichiers 2. Sélectionnez vos fichiers COBOL 3. Glissez-les sur la zone de drop 4. Relâchez

Validation automatique : - Extensions acceptées : `.cob`, `.cbl` - Taille max : 50 MB par fichier - Autres extensions : rejetées

Liste des fichiers : Chaque fichier uploadé apparaît avec : - Nom du fichier - Taille (KB/MB) - Bouton " Retirer" pour supprimer

1.8.2.3 Étape 3 : Lancer la conversion

1. Vérifiez que tous les fichiers sont listés
2. Cliquez sur le bouton **" Convertir en Spring Batch"**

Barre de progression affichée :

```
[====================60%====================]
Parsing des fichiers COBOL...
```

Étapes visibles : 1. Upload des fichiers... (20%) 2. Parsing des fichiers COBOL... (50%) 3. Génération du projet Spring Batch... (75%) 4. Téléchargement du projet... (100%)

1.8.2.4 Étape 4 : Téléchargement automatique

Dès que la conversion est terminée : - Message de succès affiché - Fichier ZIP téléchargé automatiquement - Nom : `{nom-du-projet}.zip`

Exemple : `customer-batch-migration.zip`

1.8.3 6.3 Messages et notifications

Succès :

```
 Conversion réussie!
Votre projet Spring Batch a été généré avec succès.
Le fichier customer-batch.zip a été téléchargé.
```

Erreur de validation :

```
 Erreur
Veuillez entrer un nom de projet
```

Erreur de conversion :

```
 Erreur
Conversion failed: Syntax error in customer.cob at line 42
```

1.9 7. Exemples pratiques

1.9.1 7.1 Exemple 1 : Programme COBOL simple

Fichier d'entrée : `hello.cob`

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO-WORLD.

       PROCEDURE DIVISION.
       MAIN-PARA.
           DISPLAY 'Hello from COBOL'.
           STOP RUN.
```

Conversion CLI :

```
java -jar target/cobol-translator.jar translate hello.cob \
  --package com.example.demo
```

Projet généré :

```
hello-world/
 pom.xml
 README.md
 src/
     main/
         java/com/example/demo/
            DemoApplication.java
            batch/
               HelloWorldJobConfig.java
               HelloWorldProcessor.java
            config/
```

```
            BatchConfiguration.java
        resources/
            application.properties
```

Compilation du projet généré :

```
cd hello-world
mvn clean package
mvn spring-boot:run
```

1.9.2 7.2 Exemple 2 : Programme avec données

Fichier d'entrée : `customer.cob`

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CUSTOMER-PROCESS.

       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  WS-CUSTOMER-ID      PIC 9(5).
       01  WS-CUSTOMER-NAME    PIC X(30).
       01  WS-CUSTOMER-BALANCE PIC 9(7)V99.

       PROCEDURE DIVISION.
       MAIN-PARA.
           DISPLAY 'Processing customers...'.
           STOP RUN.
```

Conversion Web : 1. Ouvrir `http://localhost:9090/conversion` 2. Nom projet : `customer-management` 3. Package : `com.bank.customer.batch` 4. Uploader `customer.cob` 5. Cliquer "Convertir"

Projet généré inclut :

```java
// CustomerEntity.java
public class CustomerEntity {
    private Integer customerId;
    private String customerName;
    private BigDecimal customerBalance;

    // Getters et Setters...
}

// CustomerProcessJobConfig.java
@Configuration
public class CustomerProcessJobConfig {
    @Bean
    public Job customerProcessJob(...) { ... }

    @Bean
    public Step customerProcessStep(...) { ... }
}
```

1.9.3 7.3 Exemple 3 : Batch de plusieurs fichiers

Fichiers d'entrée : - `customer.cob` (gestion clients) - `order.cob` (gestion commandes) - `invoice.cob` (gestion factures)

Conversion CLI par lot :

```
java -jar target/cobol-translator.jar translate-all ./cobol-batch \
  --package com.erp.batch \
  --output-dir ./java-erp-batch
```

OU Conversion Web : 1. Nom projet : `erp-batch` 2. Package : `com.erp.batch` 3. Uploader les 3 fichiers simultanément 4. Convertir

Projet généré contient : - 3 JobConfig (CustomerJobConfig, OrderJobConfig, InvoiceJobConfig) - 3 Entities (CustomerEntity, OrderEntity, InvoiceEntity) - 3 Processors - Configuration commune

---

1.10 8. Dépannage

1.10.1 8.1 Problèmes de démarrage

1.10.1.1 Problème : Port 9090 déjà utilisé Symptôme :

```
Port 9090 was already in use.
```

Solution 1 : Changer le port Éditez `src/main/resources/application.properties` :

```
server.port=8080
```

Recompilez :

```
mvn clean package
java -jar target/cobol-translator.jar
```

Solution 2 : Trouver et tuer le processus

```
# Sur Linux/macOS
lsof -ti:9090 | xargs kill -9

# Sur Windows
netstat -ano | findstr :9090
taskkill /PID <PID> /F
```

1.10.1.2 Problème : Java non trouvé Symptôme :

```
'java' is not recognized as an internal or external command
```

Solution : 1. Vérifier l'installation : `java -version` 2. Si non installé : installer Java 17 3. Ajouter au PATH (Windows)

1.10.2 8.2 Problèmes de conversion

1.10.2.1 Problème : "No valid COBOL files found" Cause : Fichiers sans extension `.cob` ou `.cbl`

Solution :

```
# Renommer les fichiers
mv program.txt program.cob
```

1.10.2.2 Problème : "Syntax error in COBOL" Cause : Fichier COBOL avec erreurs syntaxiques

Solution : 1. Vérifier le fichier COBOL original 2. Compiler avec un compilateur COBOL natif 3. Corriger les erreurs 4. Réessayer la conversion

Exemple d'erreur :

```
Syntax error at line 42:15 - mismatched input 'END'
expecting {MOVE, ADD, DISPLAY, ...}
```

→ Vérifier la ligne 42, colonne 15 du fichier

1.10.2.3 Problème : "Invalid package name" Cause : Package Java invalide

Exemples invalides : - `Com.Example` (majuscule) - `123.company` (commence par chiffre) - `my-package` (tiret interdit)

Exemples valides : - `com.example.batch` - `org.acme.migration` - `fr.company.cobol.batch`

1.10.3 8.3 Problèmes de compilation du projet généré

1.10.3.1 Problème : "BUILD FAILURE" dans le projet généré Diagnostic :

```
cd generated-project
mvn clean compile -X  # Mode debug
```

Causes courantes : 1. Java version < 17 2. Maven non configuré 3. Dépendances non téléchargées

Solution :

```
# Forcer téléchargement des dépendances
mvn dependency:resolve

# Nettoyer et recompiler
mvn clean install
```

1.10.4 8.4 Problèmes de mémoire

1.10.4.1 Problème : "OutOfMemoryError" Solution :

```
# Augmenter la mémoire JVM
java -Xmx2G -jar target/cobol-translator.jar

# Pour très gros fichiers
java -Xmx4G -jar target/cobol-translator.jar
```

1.11 9. FAQ

1.11.1 Q1 : Quels types de fichiers COBOL sont supportés ?

Réponse : Fichiers `.cob` et `.cbl` avec les 4 divisions standard (IDENTIFICATION, ENVIRONMENT, DATA, PROCEDURE).

Limitations connues : - COBOL 85 principalement - Certaines extensions vendor-specific non supportées - Copybooks : détectés mais non expansés automatiquement

1.11.2 Q2 : Le projet généré est-il prêt pour la production ?

Réponse : Le projet généré fournit une **base solide** mais nécessite généralement : - Révision de la logique métier - Ajout de tests unitaires - Configuration des sources de données - Ajustement des performances

Utilisation recommandée : - Point de départ pour migration - Prototype rapide - Analyse de faisabilité

1.11.3 Q3 : Puis-je personnaliser la génération de code ?

Réponse : Oui, plusieurs options :

1. **Templates Velocity/Freemarker** : Modifiez dans `src/main/resources/templates/`
2. **Générateurs Java** : Personnalisez dans `src/main/java/com/cobol/translator/generator/`
3. **Configuration** : Via `TranslationConfig`

1.11.4 Q4 : Comment gérer les programmes COBOL très volumineux ?

Réponse :

```
# Augmenter mémoire
java -Xmx4G -jar target/cobol-translator.jar translate large-program.cob

# OU diviser en modules plus petits
split -l 1000 large-program.cob module_
# Puis convertir chaque module
```

1.11.5 Q5 : L'interface web est-elle sécurisée ?

Réponse : L'interface actuelle est prévue pour **usage local/intranet**.

Sécurité actuelle : - Validation d'extension de fichiers - Limite de taille d'upload - Nettoyage automatique des fichiers temporaires - Pas d'authentification (ajoutez si besoin) - Pas de HTTPS par défaut (configurez si exposé)

Pour production : Ajoutez Spring Security, HTTPS, authentification.

1.11.6 Q6 : Puis-je exécuter en mode batch automatisé ?

Réponse : Oui, via CLI :

```
#!/bin/bash
# Script de conversion batch

for file in cobol-programs/*.cob; do
    java -jar cobol-translator.jar translate "$file" \
        --package com.company.batch \
```

```
        --output-dir ./converted
done
```

1.11.7 Q7 : Comment contribuer au projet ?

Réponse : 1. Fork le repository 2. Créer une branche feature 3. Développer et tester 4. Soumettre une Pull Request

Zones de contribution : - Amélioration grammaire COBOL - Nouveaux générateurs de code - Templates de projet - Tests supplémentaires

---

1.12 10. Annexes

1.12.1 10.1 Structure complète du projet

```
cobol-to-java-translator/
 pom.xml                          # Configuration Maven
 README.md                        # Documentation principale
 USER_GUIDE.md                    # Ce guide
 src/
    main/
       java/com/cobol/translator/
          ast/                 # Classes AST (49 fichiers)
          parser/              # Parsers COBOL
          generator/           # Générateurs de code
          controller/          # Contrôleurs Web
          service/             # Services métier
          model/               # Modèles de données
          config/              # Configuration
       antlr4/                  # Grammaire ANTLR4
       resources/
           templates/           # Templates HTML
           static/              # CSS, JS
           application.properties
    test/                        # Tests unitaires
 examples/                         # Fichiers COBOL d'exemple
 docs/                             # Documentation détaillée
 target/                           # Fichiers compilés

Total: ~78 fichiers Java, ~6,900 lignes de code
```

1.12.2 10.2 Ports par défaut

Service	Port	Modifiable
Interface Web	9090	Oui (application.properties)
Console H2	9090	Oui (même port que web)
Debugger Java	5005	Oui (-agentlib)

1.12.3 10.3 Variables d'environnement

```
# Port du serveur
export SERVER_PORT=8080

# Niveau de log
export LOGGING_LEVEL=DEBUG

# Répertoire temporaire
export TEMP_DIR=/tmp/cobol-translator

# Lancer avec variables
java -jar target/cobol-translator.jar
```

1.12.4 10.4 Commandes Maven utiles

```
# Compilation complète
mvn clean package

# Tests uniquement
mvn test

# Tests d'un fichier spécifique
mvn test -Dtest=CobolASTParserTest

# Générer Javadoc
mvn javadoc:javadoc

# Vérifier dépendances
mvn dependency:tree

# Analyser code (si configuré)
mvn pmd:check

# Formater code
mvn formatter:format
```

1.12.5 10.5 Raccourcis utiles

Bash/Zsh :

```
# Ajouter à ~/.bashrc ou ~/.zshrc
alias cobol-translate='java -jar ~/path/to/cobol-translator.jar translate'
alias cobol-web='java -jar ~/path/to/cobol-translator.jar'

# Utilisation
cobol-translate my-program.cob
cobol-web  # Lance l'interface web
```

1.12.6 10.6 Logs détaillés

Activer logs DEBUG :

Éditez `application.properties` :

```
logging.level.com.cobol.translator=DEBUG
logging.level.org.springframework=DEBUG
```

OU en ligne de commande :

```
java -jar target/cobol-translator.jar --debug
```

Fichier de logs :

```
# Rediriger dans un fichier
java -jar target/cobol-translator.jar > conversion.log 2>&1
```

1.12.7 10.7 Configuration avancée

application.properties complet :

```
# Application
spring.application.name=cobol-to-java-translator

# Serveur
server.port=9090
server.compression.enabled=true

# Upload
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=50MB
spring.servlet.multipart.max-request-size=100MB
spring.servlet.multipart.file-size-threshold=2MB

# Thymeleaf
spring.thymeleaf.cache=false
spring.thymeleaf.mode=HTML

# Database H2
spring.datasource.url=jdbc:h2:mem:translatordb
spring.datasource.driver-class-name=org.h2.Driver
spring.h2.console.enabled=true

# Batch
spring.batch.jdbc.initialize-schema=always
spring.batch.job.enabled=false

# Logging
logging.level.root=INFO
logging.level.com.cobol.translator=INFO
logging.pattern.console=%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
```

1.12.8 10.8 Support et contact

Documentation : - README.md - WEB_INTERFACE_README.md - INDEX_DOCUMENTATION.md

Code source : - Repository Git : [lien] - Issues : [lien]

Communauté : - Forum : [lien] - Chat : [lien]

1.13 Glossaire

ANTLR4 : ANother Tool for Language Recognition - Générateur de parser

AST : Abstract Syntax Tree - Arbre syntaxique abstrait

CLI : Command Line Interface - Interface en ligne de commande

Spring Batch : Framework Java pour traitement par lots

Maven : Gestionnaire de dépendances et build Java

Thymeleaf : Moteur de templates HTML pour Spring

H2 : Base de données en mémoire Java

JAR : Java ARchive - Archive exécutable Java

FIN DU GUIDE UTILISATEUR

Version 1.0.0 | Janvier 2026